Appendix

A Additional explanations to the guess target object

Our guessed target object is not a real object or a future detected object. The real target object will be included in one of the later sub-state-vector $\boldsymbol{s}_{o^0}$ to s_{o^n}. In our problem, initially, we do not know the objects in the environment, neither the number of objects nor their positions, which means the whereabouts of the target object are also initially unknown. To capture uncertainty about the target object compactly, we propose to "guess" the target object. It can be considered as an additional visible object, related to the target object, in transition and visual observation functions of the POMDP framework, but not detectable in the real world. It uses the same structure as the structure used to represent an object in the system, which includes an 8-cell occupancy grid map where each cell maintains a probability of whether the cell intersects with the target object. Under the POMDP framework, a belief over the guessed target object variable represents multiple guesses with different probabilities about the target object. Initially, the uncertainty about the target object can be quite large, but over time the belief in the position of the guess target object tends to converge to a probability mass whose mode is at the true target object. The basic proof of this convergency is shown as follows:

Proof. Assuming all grids are updated at least N times in the whole action sequence with $N_1 \geq N$ observations and, in each observation z_t, any $t \in [0, N_1]$, the values $\log Odd(g_i|z_t),\ i = 1, 2, 3, 4,\ g_i \in \mathcal{G}_f$ corresponding to different cases (a. the grid g_1 in field of view (FOV) where no objects are detected, b. g_2 in the FOV area where non-target objects are detected, c. g_3 in FOV where the target object is detected, d. g_4 outside the FOV area) are accurate enough to satisfy:

$$\log Odd(g_i|z_t) \begin{cases} < -\triangle < 0 & i = 1,\ 2 \\ > \triangle > 0 & i = 3 \\ = 0 & i = 4 \end{cases} \tag{1}$$

We have:

$$\log Odd(g_i|z_{1:N_1}) \begin{cases} \leq \log Odd(g_i|z_{1:N}) < -N\triangle & i = 1,\ 2 \\ \geq \log Odd(g_i|z_{1:N}) > N\triangle & i = 3 \\ = 0 & i = 4 \end{cases} \tag{2}$$

When $N \rightarrow +\infty$, we have:

$$Odd(g_i|z_{1:N_1}) \begin{cases} < \exp^{-N\triangle} \rightarrow 0 & i = 1,\ 2 \\ > \exp^{N\triangle} \rightarrow +\infty & i = 3 \\ = 1 & i = 4 \end{cases} \tag{3}$$

Introducing the relationship of Odd and probablity values $Odd(g_i|z_{1:N_1}) = \frac{P(g_i|z_{1:N_1})}{1-P(g_i|z_{1:N_1})}$, we have:

$$P(g_i|z_{1:N_1}) = \frac{Odd(g_i|z_{1:N_1})}{1 + Odd(g_i|z_{1:N_1})} \begin{cases} \rightarrow 0 & i = 1,\ 2 \\ \rightarrow 1 & i = 3 \\ = 0.5 & i = 4 \end{cases} \tag{4}$$

Normalized probabilities (0-1 range) across the grid world lead to convergence towards actual target object grid cells. □

The distribution of the guess target object (grid world) saves the belief about the target object. Intuitively, the whole grid world can be used for prediction directly. But, in POMDP problems, we need to predict the future probability of all girds in MCTS by going through the transition functions and observation functions, which will be time-consuming and memory-consuming based on the whole grid world. The application of the guess target object becomes meaningful to involve the useful information (represent grid world), is very computational (fast for transition functions and observation functions) and memory (a low-dimension vector) cheap in exploration and rollout, and is friendly for coding (use the same transition and visual observation function as the other detected real objects).

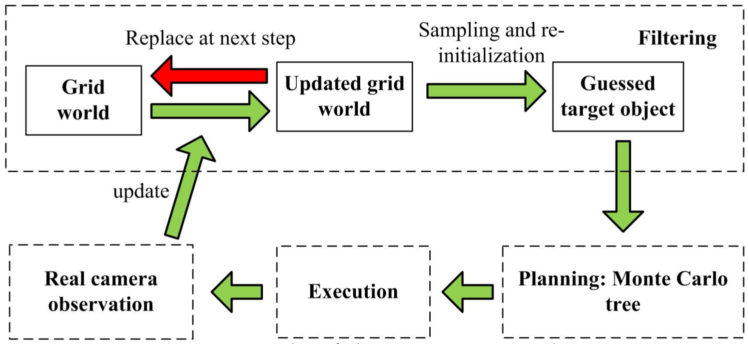


Figure 1: Role of the guess target object

The way to use the guessed target object is shown in Fig. 1. Its grid world is updated based on the real-world observation from the camera and its position will be sampled and the other terms of the guess target object will be re-initialized to use in the transition function and observation function of the Monte Carlo tree search. Then, this iterative process persists until the task is completed.

B Additional details to perform the measurement from point cloud before object detection

The main process has been stated in the article. Here, we would like to show a visible process to further explain the main process, including ICP-based scan matching, point cloud filter, point cloud segmentation, and object-oriented bounding box generation, as shown in Fig. 2.

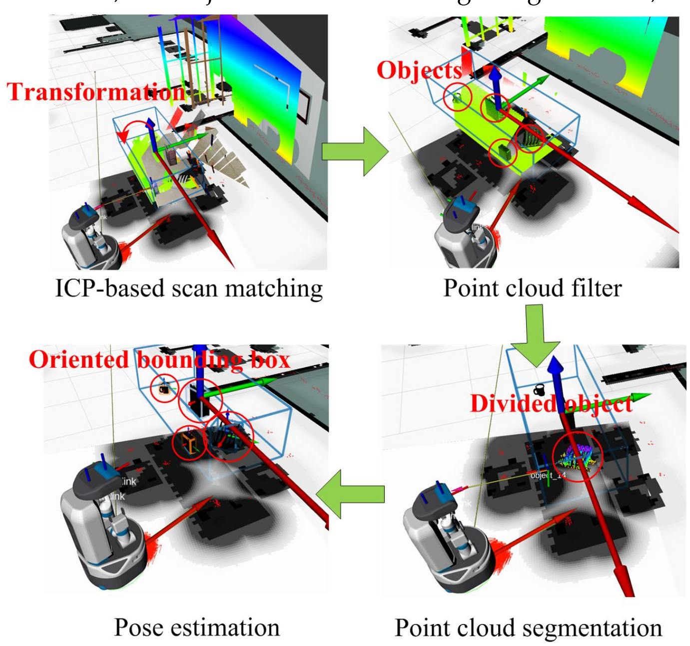


Figure 2: Measurement from point cloud

C Object detector

State-of-the-art real-time object detection systems, like YOLO, are commonly designed to divide the object into different classes and they are not matched with the target images. Meanwhile, we have the 3D point clouds of the objects, which are helpful to divide the objects in the image. So as to complete the given object detection task using several given images and some semantic words (optional), we fuse the traditional feature-matching method and YOLO toolbox to complete the object detection task.

Based on the previous point cloud segmentation, we perform it on the current visual local frame and the separated point clouds in the local frame are re-projected to the image to bound the objects in the RGB camera image forming a set of sub-images $\{\mathcal{I}_i^p, i = 1, 2, \cdots, n\}$ using the camera configuration and the perspective projection. Similar sub-images of this image $\{\mathcal{I}_i^y, i = 1, 2, \cdots, m\}$ and their corresponding semantic scores $\{s_i^y, i = 1, 2, \cdots, m\}$ can also be bounded and generated using the YOLOv5 model with pre-trained parameters. Commonly, we have $m \neq n$. A simple data association method with the nearest images and enough common areas is presented to match these two sets of sub-images. For the successful data association pair, we use the sub-image in the local frame as the image corresponding to this object. These sub-images in the detected and associated 2D boxes corresponding to different objects are matched with the target object using SIFT descriptor. The rate between the number of matched scale-invariant features and the number of all features is defined as the probability of object detection, denoted $\{s_i^d, i = 1, 2, \cdots, n\}$. If this task offers the target type, like cup and laptop, we use the mean values between the semantic scores $\{s_i^y, i = 1, 2, \cdots, n\}$ and the probability of object detection $\{s_i^d, i = 1, 2, \cdots, n\}$. The main process of the object detector is given in Fig. 3.

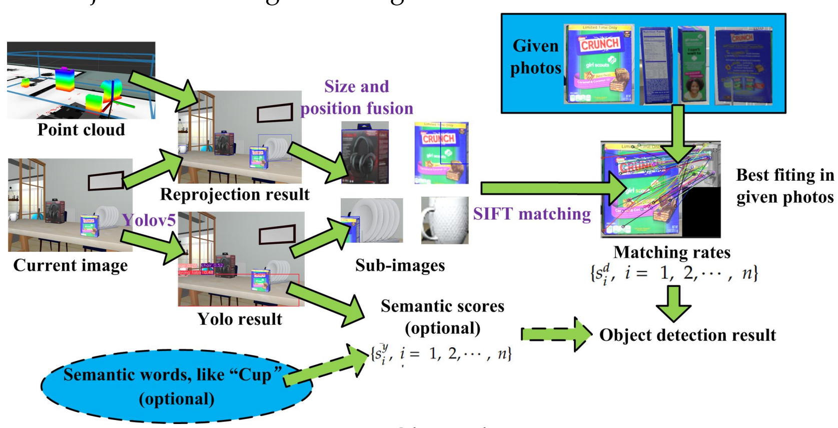


Figure 3: Object detector

D Move-ability estimation

It is easy to know that, in the real-world environment, some objects in the workspace are not moveable for the robot with a manipulator due to some physical limitations, such as the size limitation of the object, the manipulator workspace limitation, and the mobile base motion limitation. In our framework, we would like to manipulate the objects in the workspace to free some FOV, so it is better to estimate the probability of the move-ability and then update their beliefs for POMDP planning.

Based on the point cloud segmentation for the fused global point cloud, we can obtain many separated point clouds for different objects. Then, facing each point cloud in the detected frame, many candidate grasp poses are predicted by the learning-based Grasp Pose Detection (GPD) toolbox. So as to reduce the computational complexity, we select k representative grasp poses $\boldsymbol{p}_i^g,\ i = 1, 2, \cdots, k$ for each object using k-means clustering algorithm. These k grasp poses are diverse with high scores in picking success rates. The point clouds of the obstacles in the surrounding environment and these k representative grasp poses $\boldsymbol{p}_i^g$ are transformed to the local frames $T_r^g(\boldsymbol{p}_i^g)$ based on the pre-visited robot poses $\boldsymbol{p}_i^r$ during the task process. Here, it is noted that only the pre-visited robot poses are considered because the poses generated by other methods may not be reachable based on the used move-base toolbox because of the error of the AMCL localization and the complexity of the occupancy grid map. These pre-visited robot poses $\boldsymbol{p}_i^r$ are safer for implementation. Following, these transformed local poses $T_r^g(\boldsymbol{p}_i^g)$ will be set as the plan target to the robot manipulator using moveit toolbox without execution in a given time limitation t_m. The planned moveit feedback will decide the probability of this detection about the move-ability $0 < r_{move} \leq 1$ based on distance. When some objects are near the given poses within a small distance, it will be set as 1 (definitely moveable). When no solution for moveit toolbox, the move-ability r_{move} will get close to 0. In the planning stage, for each particle, we will randomly sample a random value for this object and compare it with the move-ability r_{move} to identify the move-ability in this step. Objects with too large sizes will be considered to be non-moveable $r_{move} = 0$, which is definitely not movable. An example of the candidate grasp poses is shown in Fig. 4.

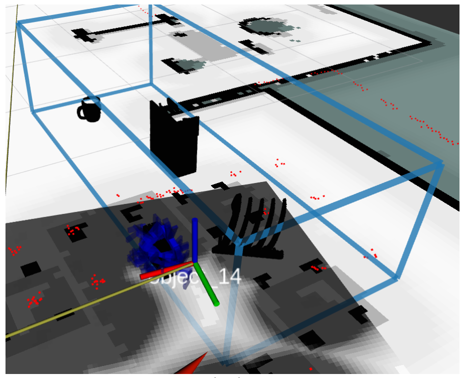


Figure 4: Candidate grasp poses

E Theorem 1

Theorem 1. *We are considering the unweighted particles for approximating the extended full-correct belief* $\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\boldsymbol{s}'_{new} = {\boldsymbol{s}^{i}_{new}}' : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})$ *and the reused approximated belief* $\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \to \boldsymbol{s}'_{add}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\boldsymbol{s}'_{new} = {\boldsymbol{s}^{i}_{new}}' : \boldsymbol{s}' \to \boldsymbol{s}'_{add})$*. Assuming the reward function is Borel measurable and bounded, denoted* $R_{max} = \|R\|_\infty$*, the belief L1 distance is limited in* $\|\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}) - \boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \to \boldsymbol{s}'_{add})\|_1 \leq \delta$*, and the optimal action selected by building the whole tree is same as the one selected by approximated value* $\max_{\boldsymbol{a}} V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}) = \max_{\boldsymbol{a}} \hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}) = \boldsymbol{a}^*$*, the optimal value function of POMDP problem using our method* $V^*(\boldsymbol{b}(\boldsymbol{s}_{new}))$ *and the optimal value function using the direct resampling way* $\hat{V}^*(\boldsymbol{b}(\boldsymbol{s}_{new}))$ *will satisfy the following bounding equation:*

$$\|V^*(\boldsymbol{b}(\boldsymbol{s}_{new})) - \hat{V}^*(\boldsymbol{b}(\boldsymbol{s}_{new}))\|_1 \leq \frac{\gamma R_{max}}{1-\gamma}\delta \tag{5}$$

Proof. Let's consider one of the new extended beliefs $\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})$. Based on the well-known α-vector, we have the optimal value of belief $\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})$ can be written as:

$$\begin{aligned}
&V^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})) \\
=&\max_{\boldsymbol{a}} Q^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}),\ \boldsymbol{a}) \\
=&\sum_{\boldsymbol{s}'_{new} \in S} \alpha(\boldsymbol{s}'_{new})\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})
\end{aligned} \tag{6}$$

where $\alpha(\boldsymbol{s}'_{new}) = R(\boldsymbol{s}'_{new},\ \boldsymbol{a}^*) + \gamma V^*(\boldsymbol{s}'_{new},\ \boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}),\ \boldsymbol{a}^*)$ and $\boldsymbol{a}^*$ is the optimal action. It is noted that the α-vector $\alpha(\boldsymbol{s}'_{new})$ is bounded by $\frac{R_{max}}{1-\gamma}$, introducing the particle representation for the belief $\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\boldsymbol{s}'_{new} = {\boldsymbol{s}^i_{new}}' : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})$, so we have:

$$
\begin{aligned}
&\|V^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})) - V^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightarrow \boldsymbol{s}'_{add}))\|_1 \\
=&\|\frac{1}{N}\sum_{i=1}^{N} \alpha(\boldsymbol{s}'_{new})\mathbb{I}(\boldsymbol{s}'_{new} = {\boldsymbol{s}^i_{new}}' : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}) - \frac{1}{N}\sum_{i=1}^{N} \alpha(\boldsymbol{s}'_{new})\mathbb{I}(\boldsymbol{s}'_{new} = {\boldsymbol{s}^i_{new}}' : \boldsymbol{s}' \rightarrow \boldsymbol{s}'_{add})\|_1 \\
\leq&\|\frac{1}{N}\sum_{i=1}^{N} \frac{R_{max}}{1-\gamma}\mathbb{I}(\boldsymbol{s}'_{new} = {\boldsymbol{s}^i_{new}}' : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}) - \frac{1}{N}\sum_{i=1}^{N} \frac{R_{max}}{1-\gamma}\mathbb{I}(\boldsymbol{s}'_{new} = {\boldsymbol{s}^i_{new}}' : \boldsymbol{s}' \rightarrow \boldsymbol{s}'_{add})\|_1 \\
=&\frac{R_{max}}{1-\gamma}\|\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}) - \boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightarrow \boldsymbol{s}'_{add})\|_1 \\
\leq&\frac{R_{max}}{1-\gamma}\delta
\end{aligned}
\tag{7}
$$

Based on

$$
\max_{\boldsymbol{a}\in A}\left(\sum_{\boldsymbol{s}_{new}\in S} R(\boldsymbol{s}_{new}, \boldsymbol{a}, \boldsymbol{s}'_{new})\boldsymbol{b}(\boldsymbol{s}_{new}) + \gamma\sum_{o\in O}\sum_{\boldsymbol{s}'_{new}\in S} Z(\boldsymbol{s}'_{new}, \boldsymbol{a}, \boldsymbol{o}')T(\boldsymbol{s}_{new}, \boldsymbol{a}, \boldsymbol{s}'_{new})V^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add}))\right)
\tag{8}
$$

and the optimal action $\boldsymbol{a}^*$, we have:

$$
\begin{aligned}
&\|V^*(\boldsymbol{b}(\boldsymbol{s}_{new})) - \hat{V}^*(\boldsymbol{b}(\boldsymbol{s}_{new}))\|_1 \\
=&\Bigg\|\sum_{\boldsymbol{s}_{new}\in S} R(\boldsymbol{s}_{new}, \boldsymbol{a}^*, \boldsymbol{s}'_{new})\boldsymbol{b}(\boldsymbol{s}_{new}) + \gamma\sum_{o\in O}\sum_{\boldsymbol{s}'_{new}\in S} Z(\boldsymbol{s}'_{new}, \boldsymbol{a}^*, \boldsymbol{o}')\ T(\boldsymbol{s}_{new}, \boldsymbol{a}^*, \boldsymbol{s}'_{new})V^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})) \\
&- \sum_{\boldsymbol{s}_{new}\in S} R(\boldsymbol{s}_{new}, \boldsymbol{a}^*, \boldsymbol{s}'_{new})\boldsymbol{b}(\boldsymbol{s}_{new}) - \gamma\sum_{o\in O}\sum_{\boldsymbol{s}'_{new}\in S} Z(\boldsymbol{s}'_{new}, \boldsymbol{a}^*, \boldsymbol{o}')\ T(\boldsymbol{s}_{new}, \boldsymbol{a}^*, \boldsymbol{s}'_{new})V^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightarrow \boldsymbol{s}'_{add}))\Bigg\|_1 \\
=&\Bigg\|\gamma\sum_{o\in O}\sum_{\boldsymbol{s}'_{new}\in S} Z(\boldsymbol{s}'_{new}, \boldsymbol{a}^*, \boldsymbol{o}')T(\boldsymbol{s}_{new}, \boldsymbol{a}^*, \boldsymbol{s}'_{new})\ V^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightleftarrows \boldsymbol{s}'_{add})) - \gamma\sum_{o\in O}\sum_{\boldsymbol{s}'_{new}\in S} Z(\boldsymbol{s}'_{new}, \boldsymbol{a}^*, \boldsymbol{o}') \\
&\quad T(\boldsymbol{s}_{new}, \boldsymbol{a}^*, \boldsymbol{s}'_{new})V^*(\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightarrow \boldsymbol{s}'_{add}))\Bigg\|_1 \\
\leq&\gamma\sum_{o\in O}\sum_{\boldsymbol{s}'_{new}\in S} Z(\boldsymbol{s}'_{new}, \boldsymbol{a}^*, \boldsymbol{o}')T(\boldsymbol{s}_{new}, \boldsymbol{a}^*, \boldsymbol{s}'_{new})\frac{R_{max}}{1-\gamma}\delta \\
=&\frac{\gamma R_{max}}{1-\gamma}\delta
\end{aligned}
\tag{9}
$$

The proof is completed. □

F Corollary 2

Corollary 1. *If the distance between the value function for the optimal action $V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*)$ and the value function for any sub-optimal action $V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*_{sub})$ is larger than $\frac{2\gamma R_{max}}{1-\gamma}\delta$, the optimal action obtained by the tree reuse approximation way will be same as the optimal action using the direct resampling way* $\max_{\boldsymbol{a}} V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}) = \max_{\boldsymbol{a}} \hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}) = \boldsymbol{a}^*$.

Proof. Based on Corollary 1, we have:

$$
\hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*_{sub}) \leq V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*_{sub}) + \frac{\gamma R_{max}}{1-\gamma}\delta
\tag{10}
$$

and

$$V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*) - \frac{\gamma R_{max}}{1-\gamma}\delta \leq \hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*) \tag{11}$$

The solution does not change, which means that the approximated value function using sub-optimal action $\boldsymbol{a}_{sub}$ is smaller than the one with optimal action, satisfying $\hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*) > \hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*_{sub})$. Here, considering Eq. (13) and Eq. (12), if the upper bound of $\hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*_{sub})$ is smaller than the lower bound of $\hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*)$, satisfying:

$$\begin{aligned} &V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*_{sub}) + \frac{\gamma R_{max}}{1-\gamma}\delta < V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*) - \frac{\gamma R_{max}}{1-\gamma}\delta \\ &\Rightarrow V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*) - V(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*_{sub}) > \frac{2\gamma R_{max}}{1-\gamma}\delta, \end{aligned} \tag{12}$$

We have:

$$\hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*_{sub}) < \hat{V}(\boldsymbol{b}(\boldsymbol{s}_{new}), \boldsymbol{a}^*) \tag{13}$$

The proof is completed. □

G Solver summary

The whole online GPOMDP solver for our object search is shown in the following pseudocode Algorithms 1 to 7. Our proposed method follows the common procedure with four alternating stages, planning, execution, obtaining observation, and filtering shown in Algorithms 1. The optimal action for the current belief is selected based on my solver proposed in Section 5 with belief tree reuse shown in Algorithms 2. Two key points should be pointed out. The first one is that, because of the observation introduced by added objects, after the state update in Section 5 Step 2, some observation nodes will be decoupled into multiple new observation nodes. Hence, the observation identification needs to be performed in line 14 Algorithm 3. The other point is that we save the observation in two hash tables, one from observation to identification ID and the other one from identification ID to observation.

Algorithm 1 Proposed GPOMDP method with growing state space

Input: POMDP 8-tuple $< \mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \boldsymbol{b}_0, \gamma >$: state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, transition function $T(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$, observation function $O(\boldsymbol{s}', \boldsymbol{a}, \boldsymbol{o})$, reward function $R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}')$, initial belief $\boldsymbol{b}_0$, and discount factor γ; and Communicable Robot.
Output: Optimal action sequence for this POMDP problem.

```
Sampling initial state {s_i}_{i=1, 2, ..., N} based on initial belief b_0, where is generated based on grid world
    of the fake object.
while True do
    a* ← PLANNING(b)
    Communicate with the robot, execute a*, and reach an unknown state s_unknown
    o ← OBSERVATION(s_unknown)
    Update the belief b with a* and o using Filters
    if The object searching task is completed then
        break
    end if
end while
```

Algorithm 2 *PLANNING*($\boldsymbol{b}$)

Input: Current belief $\boldsymbol{b}$
Output: Optimal action $\boldsymbol{a}^*$ in this step

if New objects $\boldsymbol{s}'_{add}$ are detected and old tree $\mathcal{T}$ exists **then**
  Cut some history from history List H based on previous optimal action $\boldsymbol{a}^*$ and obtained observation $\boldsymbol{o}$
  Extend the action space $\boldsymbol{a}_{add}$ for the newly detected objects including removing and declaring actions
  for all particles in parallel **do**
    $\boldsymbol{s}'_{add} \leftarrow T(\boldsymbol{s}_{add}, \boldsymbol{a})$
    $\boldsymbol{o}_{add} \leftarrow Z(\boldsymbol{s}'_{add}, \boldsymbol{a})$
    $H' \leftarrow H' \cup \{(\boldsymbol{s}', \boldsymbol{s}'_{add}), \boldsymbol{a}, \boldsymbol{o}_{add}\}$
  end for ▷ Update the state vector and observations of the particles considering the newly detected objects. Here we need to consider the effect from the old object to the newly detected object but do not consider the reverse effect. $\boldsymbol{b}(\boldsymbol{s}'_{new} : \boldsymbol{s}' \rightarrow \boldsymbol{s}'_{add})$
  for all particles within updated history list $\{h'\} \in H'$ **do** ▷ Generate the belief tree $\mathcal{T}'(\boldsymbol{s})$ using the updated history List H'.
    SIMULATE_NEW_OBJECTS(H', $\varnothing$, *particle*, *depth*)
  end for
end if
while time permitting or particle number limitation **do** ▷ Sampling more particles for MCTS if the reuse tree operation does not spend too much planning time.
  Sampling a state $\boldsymbol{s}_{new}$ based on belief $\boldsymbol{b}$
  SIMULATE($\boldsymbol{s}_{new}$, $\varnothing$, *depth*)
end while
return $\arg\max_{\boldsymbol{a} \in \mathcal{A}} \hat{Q}(\boldsymbol{b}, \boldsymbol{a})$

Algorithm 3 *SIMULATE_NEW_OBJECTS*(H', h, i, j)

Input: The record of all history H' with saved states, actions, observations, and reward; The particle id i; The depth j
Output: Discounted total reward r

$h' \leftarrow H'(i, 0 : j)$ ▷ Follow the old history.
if $j == |h'|_{rollout}$ **then**
  if Not Action node $T(h' + \{\boldsymbol{a}\})$ **then**
    for all $\boldsymbol{a} \in \mathcal{A}$ **do**
      Action node $T(h' + \{\boldsymbol{a}\}) \leftarrow (N(h' + \{\boldsymbol{a}\}), V(h' + \{\boldsymbol{a}\}), \varnothing)$
    end for
  end if
  return *ROLLOUT_NEW_OBJECTS*(h', $j + 1$)
else
  if not $T(h' + \{\boldsymbol{a}\})$ **then**
    for all $\boldsymbol{a} \in \mathcal{A}$ **do**
      Action node $Node(h' + \{\boldsymbol{a}\}) \leftarrow (N(h' + \{\boldsymbol{a}\}), V(h' + \{\boldsymbol{a}\}), \varnothing)$
    end for
    Action node $Node(h' + \{\boldsymbol{a}\}) \leftarrow (N(h' + \{\boldsymbol{a}\}), V(h' + \{\boldsymbol{a}\}), \varnothing)$
  end if
  $\{\boldsymbol{s}_{new}, \boldsymbol{s}'_{new}, \boldsymbol{a}, \boldsymbol{o}_{add}, R(\boldsymbol{s}_{new}, \boldsymbol{a}, \boldsymbol{s}'_{new})\} \leftarrow H'(i, j)$
  if $j < |h'|_{rollout}$ **then**
    $r \leftarrow \gamma SIMULATE\_NEW\_OBJECTS(H', h + \{\boldsymbol{a}, \boldsymbol{o}_{add}\}, i, j + 1) + R(\boldsymbol{s}_{new}, \boldsymbol{a}, \boldsymbol{s}'_{new})$
    $B(h') \leftarrow B(h') \cup \{\boldsymbol{s}_{new}\}$
    $N(h') \leftarrow N(h') + 1$
    $N(h' + \{\boldsymbol{a}\}) \leftarrow N(h' + \{\boldsymbol{a}\}) + 1$
    $\hat{Q}(h' + \{\boldsymbol{a}\}) \leftarrow \hat{Q}(h' + \{\boldsymbol{a}\}) + \frac{r - \hat{Q}(h' + \{\boldsymbol{a}\})}{N(h' + \{\boldsymbol{a}\})}$ ▷ Monte Carlo update.
  end if
end if

H Discussion about different resolution for grid world

The grid world's finer resolution offers a unique advantage in enhancing the reliability of probability updating within the grid. This advantage becomes particularly pronounced when dealing with smaller target objects. It's important to acknowledge that opting for this finer resolution does entail a slightly higher computational complexity (odds updating is

Algorithm 4 *SIMULATE*($\boldsymbol{s}$, h, $depth$)

Input: A given state $\boldsymbol{s}_{new}$, the previous history h, the node depth $depth$
Output: Discounted total reward r

Observation node $Node(h) \leftarrow (N(h),\ V(h),\ depth)$
if Not $Node(ha)$ **then**
  for all $\boldsymbol{a} \in \mathcal{A}$ **do**
    Action node $Node(h + \{\boldsymbol{a}\}) \leftarrow (N(h + \{\boldsymbol{a}\}),\ V(h + \{\boldsymbol{a}\}),\ depth)$
  end for
  return $ROLLOUT(h', j + 1)$
else
  $\boldsymbol{a} \leftarrow \mathrm{argmax}_{\boldsymbol{a}} C_b \sum_{o \in O} Z(\boldsymbol{s}'_{new}, \boldsymbol{a}, \boldsymbol{o}') \| \boldsymbol{b}((\boldsymbol{s}',\ \boldsymbol{s}'_{add}) : s' \rightarrow \boldsymbol{s}'_{add}) - \boldsymbol{b}((\boldsymbol{s}',\ \boldsymbol{s}'_{add}) : s' \rightleftarrows \boldsymbol{s}'_{add}) \|_1 + V^*(\boldsymbol{b}, \boldsymbol{a}) + C_N \sqrt{\frac{logN(b)}{N(\boldsymbol{b},\boldsymbol{a})}}$ ▷ Novel UCB strategy.
  $\boldsymbol{s}'_{new} \leftarrow T(\boldsymbol{s}_{new}, \boldsymbol{a})$
  $\boldsymbol{o}' \leftarrow Z(\boldsymbol{s}'_{new}, \boldsymbol{a})$
  if $\boldsymbol{s}'_{new}$ is not terminal state **then**
    $r \leftarrow \gamma SIMULATE(\boldsymbol{s}_{new},\ h + \{\boldsymbol{a},\ \boldsymbol{o}_{add}\},\ depth + 1) + R(\boldsymbol{s}_{new},\ \boldsymbol{a},\ \boldsymbol{s}'_{new})$
    $B(h') \leftarrow B(h') \cup \{\boldsymbol{s}_{new}\}$
    $N(h') \leftarrow N(h') + 1$
    $N(h' + \{\boldsymbol{a}\}) \leftarrow N(h' + \{\boldsymbol{a}\}) + 1$
    $\hat{Q}(h' + \{\boldsymbol{a}\}) \leftarrow \hat{Q}(h' + \{\boldsymbol{a}\}) + \frac{r - \hat{Q}(h' + \{\boldsymbol{a}\})}{N(h' + \{\boldsymbol{a}\})}$ ▷ Monte Carlo update.
  end if
end if

Algorithm 5 *ROLLOUT_NEW_OBJECTS*(h', j)

Input: The history h, the depth j
Output: Discounted total roll-out reward

if $j > |h'|$ **then**
  return 0
end if
$\{\boldsymbol{s}_{new}, \boldsymbol{s}'_{new},\ \boldsymbol{a},\ \boldsymbol{o}_{add},\ R(\boldsymbol{s}_{new},\ \boldsymbol{a},\ \boldsymbol{s}'_{new})\} \leftarrow h'(j)$
if $j == |h'|$ **then**
  return $R(\boldsymbol{s}_{new},\ \boldsymbol{a},\ \boldsymbol{s}'_{new})$
end if
return $r + \gamma ROLLOUT\_NEW\_OBJECTS(h' + \{\boldsymbol{a},\ \boldsymbol{o}_{add}\}, j + 1)$

Algorithm 6 *ROLLOUT*($\boldsymbol{s}, h, j$)

Input: The history h, the depth j, the current state s
Output: Discounted total roll-out reward

if $\gamma^j < C_r$ or $j > C_{max}$ **then** ▷ Update the state vector and observations of the particles considering the newly detected objects. Here we need to
  return 0
end if
$\boldsymbol{a} \leftarrow random(\mathcal{A})$
$\boldsymbol{s}' \leftarrow T(\boldsymbol{s}, \boldsymbol{a})$
$\boldsymbol{o} \leftarrow Z(\boldsymbol{s}', \boldsymbol{a})$
$r \leftarrow R(\boldsymbol{s},\ \boldsymbol{a},\ \boldsymbol{s}')$
return $r + \gamma ROLLOUT(\boldsymbol{s},\ h + \{\boldsymbol{a},\ \boldsymbol{o}\}, j + 1)$

very cheap for large datasets). Upon conducting tests, it becomes evident that the good grid resolution should be smaller than the minimum dimensions of the target object along both x and y axes. We are here to correct a typo in our previous version. All the experiments are implemented in 2 cm resolution instead of 5 cm. For comparison, we present statistical results for the scenario using 10 cm resolution: 24803.5±3792.3|39.9±6.9|60%. The result is poorer than the POMCP method without using fake objects in success rate. To dissect this phenomenon, we manually executed a designated sequence of actions: $\imath move\_head\_5 - move\_base_0 - move\_base\_1 - move\_base\_2 - move\_base\_3 - move\_lift\_2 - move\_base\_0 - move\_base\_1 - move\_base\_2 - move\_base\_3 - remove\_object\_4 - move\_head\_8 - move\_base\_0 - move\_base\_1 - move\_base\_2 - move\_base\_3\jmath$ for our method with 2 cm and 10 cm resolution, and the resulting changes in probability distributions across the grid world are visually depicted in Fig. 5. The salient observation is that when the resolution is relatively

Algorithm 7 $\boldsymbol{o} \leftarrow OBSERVATION(\boldsymbol{s}_{unknown})$

Input: Current configuration including robot and environment
Output: Obtained observation for pose and objects

1: ICP-based scan matching to get the rigid transformation T_r between AMCL pose estimation and 3D point cloud map
2: Point cloud fusion $\mathcal{P}_j = \mathcal{M} \bigcup \mathcal{F}_0 \bigcup \cdots \bigcup \mathcal{F}_j$
3: Point cloud filter to remove the point cloud outside the workspace
4: Point cloud segmentation to divide the point cloud into multiple point clouds $\{o_0, o_1, \cdots, o_n\} \in \mathcal{M}'_j$
5: Minimal oriented bounding box estimation for the poses $\{p^o_0, p^o_1, \cdots, p^o_n\} \in \mathcal{M}'_j$ of different point clouds
6: SIFT and YOLO-based object detector with detected scores $\{(s^y_i + s^d_i)/2, i = 1, 2, \cdots, n\}$
7: Move ability estimation based on GPD toolbox
8: Data association and summarize all measurements, including robot pose estimation, object pose and size estimation, probability of object detection (most important, only used in the belief tree), and move-ability estimation

large, grid probability updates lose precision, particularly when the target object is situated close to an obstacle. The use of a larger resolution not only fails to enhance our method's performance but also introduces potential distortions in the information of the fake object. In short, we recommend the used resolution had better be smaller than the object sizes to improve the accuracy of updating odds values.

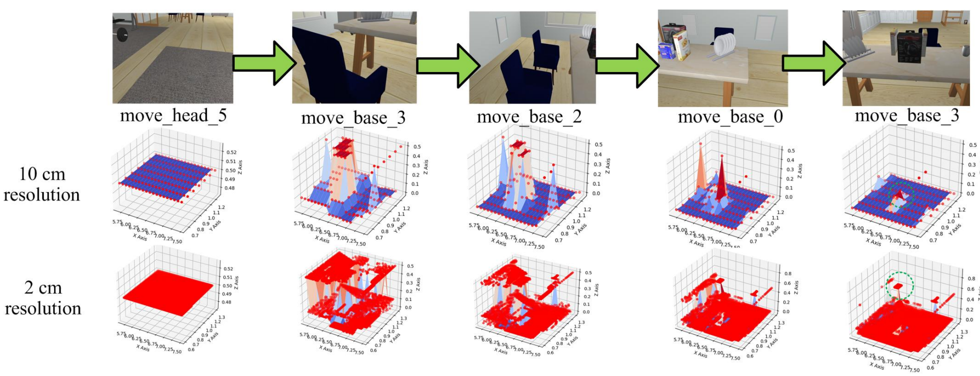


Figure 5: Candidate grasp poses

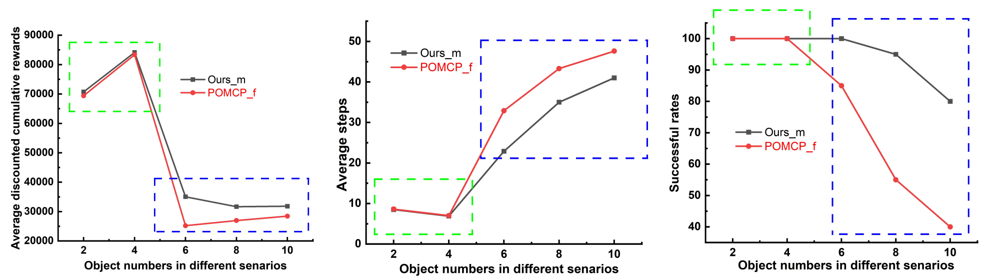
I More simulation results with different object numbers, reward settings, and thresholds

Indeed, the computational complexity of our framework will mainly increase with the number of updating objects. These objects play a crucial role in both transition and visual observation functions, causing fewer sampled particles. Consequently, the computational expenditure associated with these declared objects becomes relatively economical, as it no longer necessitates exhaustive testing of occlusion relationships among all other objects. The good news is that commonly the number of the updating objects is limited by the declaring action. To facilitate a clearer understanding of how our method performs under varying object counts, we have conducted an ablation study. This study includes a comprehensive set of experiments, encompassing an increasing number of objects ranging from 2 to 10, with increments of 2. The scenarios, as depicted in Fig. 6, comprise 20 trials each, all adhering to a consistent planning time limit of 60 seconds per step. The statistical results are reported in Fig. 7. Our method will perform obviously better, if the object number is relatively large (blue box areas), because we reuse the useful belief tree and avoid branch-cutting caused by the newly detected objects, which is more common in scenarios with more objects.

While some parameters of our framework require manual configuration, it's worth noting that the majority of these parameters do not significantly impact the final performance of the framework. Most of them are not sensitive to the final performance. As an example, the performance does not change too much, if orders of magnitude between R_{max} and R_{min} satisfy $R_{max} >> R_{min}$. For instance, R_{min} varying from -1 to -20 does not exert a substantial influence

Object number = 2 **Object number = 4** **Object number = 6** **Object number = 8** **Object number = 10**

Figure 6: Scenarios with different object numbers from 2 to 10.



(a) Average discounted cumulative rewards (larger is better) (b) Average steps (fewer is better) (c) Successful rate (higher is better)

Figure 7: The comparisons between Ours_m with POMCP_f.

on performance outcomes. To elucidate this phenomenon, we present a comparison between the results using $R_{min} = -20$ (the case in our paper) and for the $Hidden_1$ case in Table 1.

Table 1: 95% confidence interval of discounted cumulative reward, steps, and successful rate (within 50 steps)

Scenarios	$R_{min} = -1$	$R_{min} = -20$
$Hidden_1$	55574.8 ± 6225.3 | 15.3 ± 2.0 | 100%	54351.1 ± 7180.3 | 15.9 ± 2.2 | 100%

Few parameters may affect the performance a lot. For example, before successfully declaring, we need to update the belief of the log-odds of 8 grid values and then complete the declaring action by comparing the mean value of the minimal $n_{odds} = 2$ log-odds with the thresholds C_d^o for the obstacle object and C_t^o for the target object. The smaller n_{odds} means that this condition is hard to be satisfied and we need to observe more directions of the objects. As n_{odds} decreases, the removing and declaration actions become more dependable, owing to the augmented array of diverse observations derived from different orientations. Our method retains more useful branches after receiving observations and it will show obvious advantages when the target belief is harder to reach. With this goal, we increase n_{odds} to 4 and 6 and show the statistical results of the scenario with 6 objects (Fig. 6) in Table 2 with 20 trials.

Table 2: 95% confidence interval of discounted cumulative reward, steps, and successful rate (within 50 steps)

Scenarios	$Ours_b$	$POMCP_f$
2	**35038.7 ± 4604.5** | **22.9 ± 2.5** | **100%**	25192.4 ± 3045.1 | 32.9 ± 3.5 | 85%
4	45327.6 ± 4491.9 | **19.3 ± 1.8** | **100%**	**47258.1 ± 6185.9** | 20.2 ± 3.8 | 90%
6	**72967.1 ± 10896.3** | 12.1 ± 2.8 | **100%**	72081.7 ± 9524.8 | **12.0 ± 2.7** | **100%**